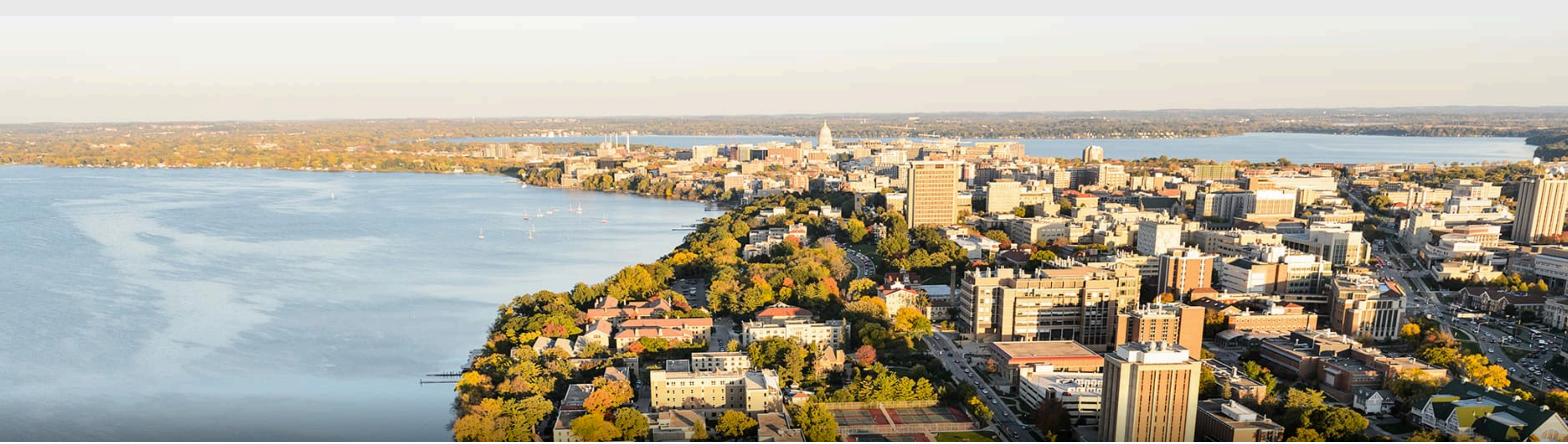


CS 760: Machine Learning
Linear Regression & Logistic Regression

Kirthi Kandasamy

University of Wisconsin-Madison

February 15, 2023

Outline

- **Linear Regression**
  - Gradient-descent based solutions
- **Logistic Regression**
  - Maximum likelihood estimation, setup, comparisons
- **Logistic Regression: Multiclass**
  - Extending to multiclass, softmax, cross-entropy
- **Gradient Descent & SGD**
  - Convergence proof for GD, introduction to SGD

Outline

- **Linear Regression**
  - Gradient-descent based solutions
- **Logistic Regression**
  - Maximum likelihood estimation, setup, comparisons
- **Logistic Regression: Multiclass**
  - Extending to multiclass, softmax, cross-entropy
- **Gradient Descent & SGD**
  - Convergence proof for GD, introduction to SGD

Linear Regression: Setup

- **Training**: Given a dataset, where $x^{(i)} \in \mathbb{R}^d, y^{(i)} \in \mathbb{R}$

$$\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \ldots, (x^{(m)}, y^{(m)})\}$$

  - We will assume, $x_1^{(i)} = 1$ for all $i \in \{1, \ldots, m\}$

- Find $f_\theta(x) = \theta^T x = \sum_{i=1}^{d} \theta_i x_i$ which minimizes

Loss function

$$\ell(f_\theta) = \frac{1}{n} \sum_{j=1}^{n} (f_\theta(x^{(j)}) - y^{(j)})^2$$

Hypothesis Class

Linear Regression: Normal equations

- **Set gradient to 0 w.r.t. the weight,**

$$\nabla \ell(f_\theta) = \nabla \frac{1}{n} \|X\theta - y\|_2^2 = 0$$
$$\implies \nabla[(X\theta - y)^T (X\theta - y)] = 0$$
$$\implies \nabla[\theta^T X^T X\theta - 2\theta^T X^T y + y^T y] = 0$$
$$\implies 2X^T X\theta - 2X^T y = 0$$
$$\implies \theta = (X^T X)^{-1} X^T y$$

Regularized variants

Ridge regression:

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_2^2$$

Lasso regression:

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_1$$

Iterative Methods: Gradient Descent

- What if there's no closed-form solution?
- Use an iterative approach. Goal: get closer to solution.

- Gradient descent.
  - Suppose we're computing $\min_{\theta} g(\theta)$
  - Start at some θ_0
  - Iteratively compute $\theta_{t+1} = \theta_t - \alpha \nabla g(\theta_t)$
  - Stop after some # of steps

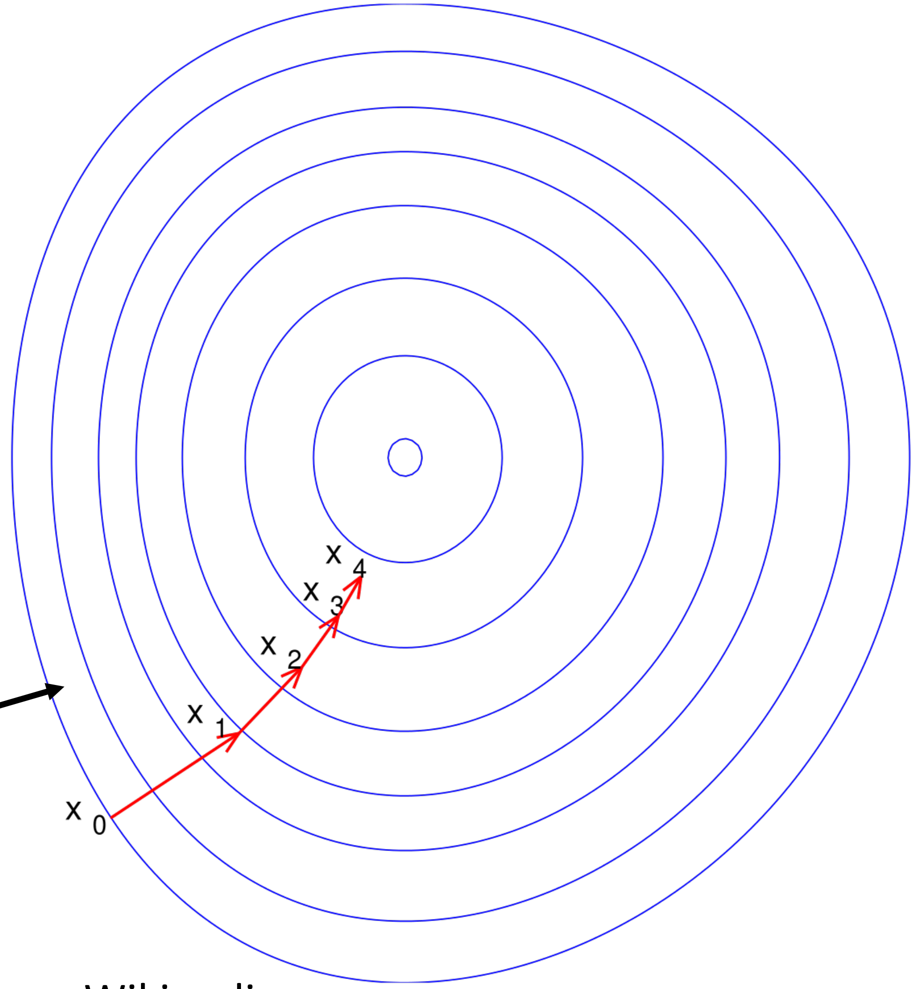
Learning rate/step size

Gradient Descent: Illustration

- **Goal**: steps get closer to minimizer

- Some notes:
  - Step size can be fixed or a function

  - Under certain conditions, will converge to global minimum
    - Need **convexity** for this



Level Sets

Wikipedia

Gradient Descent: Linear Regression

- Back to our linear regression problem.

- Want to find $\min_{\theta} \ell(f_\theta) = \min_{\theta} \frac{1}{n} \|X\theta - y\|_2^2$
  - What is our gradient? $\nabla \ell(f_\theta) = \frac{1}{n}(2X^T X\theta - 2X^T y)$
  - So, plugging in , we get

$$\theta_{t+1} = \theta_t - \alpha \frac{1}{n}(2X^T X\theta_t - 2X^T y)$$

Linear Regression: Normal Equations vs GD

- Let us compare **computation costs.**
- Normal Equations
  - Check dimensions

$$\theta = (\underbrace{X^T X}_{d \times d})^{-1} \underbrace{X^T}_{d \times n} \underbrace{y}_{n \times 1}$$

  - Cost: (i) invert matrix, $\Theta(d^3)$. (ii) multiplication, $\Theta(d^2 n)$.
  - **Total**: $\Theta(d^2 n + d^3)$.

Recall: by standard methods, inverting a square *m* x *m* matrix is $\Theta(m^3)$.

Multiplying a *m* x *p* with a *p* x *q* matrix is $\Theta(mpq)$

Linear Regression: Normal Equations vs GD

- Let us compare **computation costs**.
- Normal Equations $\theta = (X^T X)^{-1} X^T y$
  - **Total Cost**: $\Theta(d^2n + d^3)$.

- Gradient Descent: t iterations

$$\theta_{t+1} = \theta_t - \alpha \frac{1}{n}(2X^T X\theta_t - 2X^T y)$$

  - Cost: $\Theta(dn)$ at each step.
  - **Total Cost**: $\Theta(dnt)$.

If we do "few" steps t, then **GD is cheaper**: $t<\max\{d, d^2/n\}$

Break & Quiz

Q: Suppose you find that your linear regression model is under fitting the data. In such situation which of the following options would you consider?

A. *Add more variables*
B. *Start introducing polynomial degree variables*
C. *Use L1 regularization*
D. *Use L2 regularization*

1. A, B, C
2. A, B, D
3. A, B
4. A, B, C, D

Q: Suppose you find that your linear regression model is under fitting the data. In such situation which of the following options would you consider?

A. *Add more variables*
B. *Start introducing polynomial degree variables*
C. *Use L1 regularization*
D. *Use L2 regularization*

1. A, B, C
2. A, B, D
3. A, B
4. A, B, C, D

In case of under fitting, you need to induce more variables in variable space or you can add some polynomial degree variables to make the model more complex to be able to fit the data better. Regularization is unlikely to help. Regularization is typically used in case of overfitting.

Q: How do you choose the regularization paramater λ in ridge/lasso regression?

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_2^2$$

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_1$$

Q: How do you choose the regularization paramater λ in ridge/lasso regression?

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_2^2$$

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2 + \lambda\|\theta\|_1$$

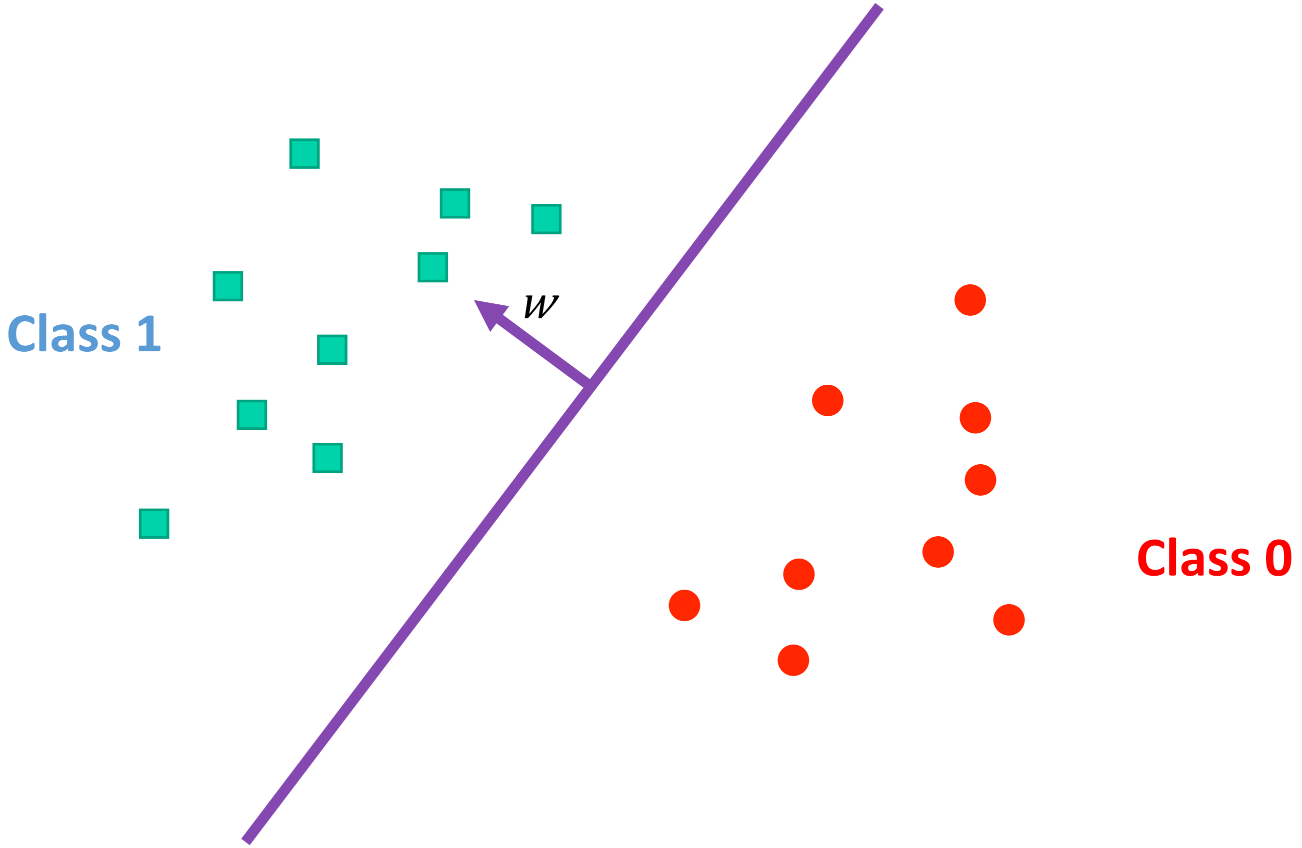
Ans: tuning (validation) set, cross validation etc.

Outline

- **Linear Regression**
  - Gradient-descent based solutions
- **Logistic Regression**
  - Maximum likelihood estimation, setup, comparisons
- **Logistic Regression: Multiclass**
  - Extending to multiclass, softmax, cross-entropy
- **Gradient Descent & SGD**
  - Convergence proof for GD, introduction to SGD

Classification: Linear

- We've been talking about regression. What about classification with linear models?



Linear Classification: Attempt 1

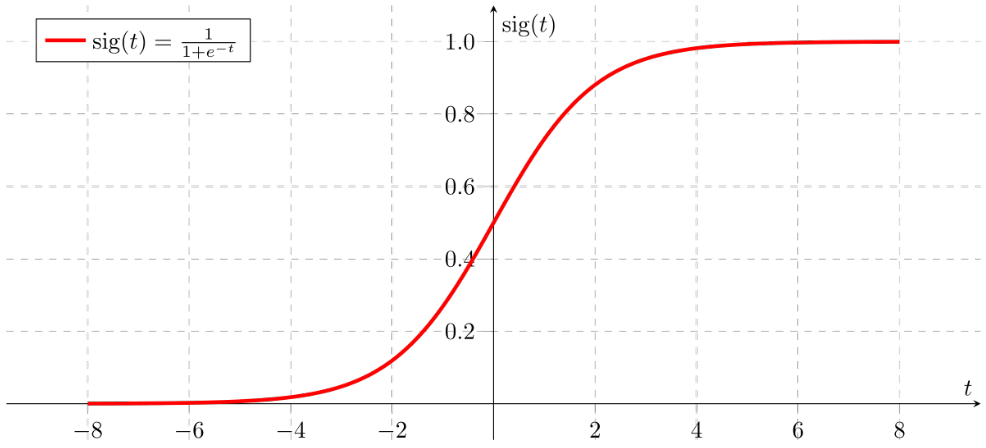
- Hyperplane: solutions to $\theta^T x = c$
  - note: d-1 dimensional
- So… try to use such hyperplanes as separators?
  - Model: $f_\theta(x) = \theta^T x$
  - Predict: y=1 if $\theta^T x > 0$, y=0 otherwise?
  - I.e, $y = \text{step}(f_\theta(x))$
  - Train: 0/1 loss, or, $$\ell(f_\theta) = \frac{1}{m}\sum_{i=1}^{m} 1\{\text{step}(f_\theta(x^{(i)}) \neq y^{(i)})$$

Difficult to optimize!!

Linear Classification: Attempt 2

- Let us think probabilistically. Learn $P_\theta(y|x)$ instead

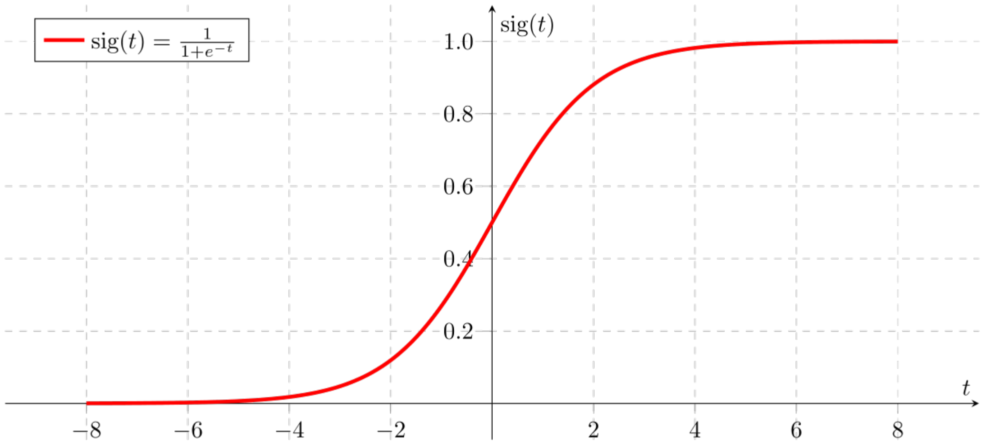
- How?
  - Specify the conditional distribution $P_\theta(y|x)$
  - Use maximum likelihood estimation (MLE) to derive a loss
  - Run gradient descent (or related optimization algorithm)



Linear Classification: Attempt 2

- Let us think probabilistically. Learn $P_\theta(y|x)$ instead

- How?
  - Specify **the conditional distribution** $P_\theta(y|x)$
  - Use **maximum likelihood estimation (MLE)** to derive a loss
  - Run gradient descent (or related optimization algorithm)



Digression: Maximum Likelihood Estimation

Likelihood function

- Captures the probability of seeing some data as a function of model parameters:

$$\mathcal{L}(\theta; X) = P_\theta(X)$$

  - If data is iid, we have $\mathcal{L}(\theta; X) = \prod_j p_\theta(x_j)$

- Often more convenient to work with the log likelihood
  - Log is a monotonic + strictly increasing function

Maximum Likelihood

- For some set of data, find the parameters that maximize the likelihood / log-likelihood

$$\hat{\theta} = \arg\max_{\theta} \mathcal{L}(\theta; X)$$

- Example: suppose we have n samples from a Bernoulli distribution

$$P_\theta(X = x) = \begin{cases} \theta & x = 1 \\ 1 - \theta & x = 0 \end{cases}$$

Then, if k of the n samples are 1

$$\mathcal{L}(\theta; X) = \prod_{i=1}^{n} P(X = x_i) = \theta^k (1-\theta)^{n-k}$$

Maximum Likelihood: Example

- Want to maximize likelihood w.r.t. Θ

$$\mathcal{L}(\theta; X) = \prod_{i=1}^{n} P(X = x_i) = \theta^k (1-\theta)^{n-k}$$

- Differentiate (use product rule) and set to 0. Get

$$\theta^{h-1}(1-\theta)^{n-h-1}(h - n\theta) = 0$$

- So: ML estimate is $\hat{\theta} = \frac{h}{n}$

h = |{x_i | x_i = 1}|

ML: Conditional Likelihood

- Similar idea, but now using conditional probabilities:

$$\mathcal{L}(\theta; Y, X) = p_\theta(Y|X)$$

  - If data is iid, we have

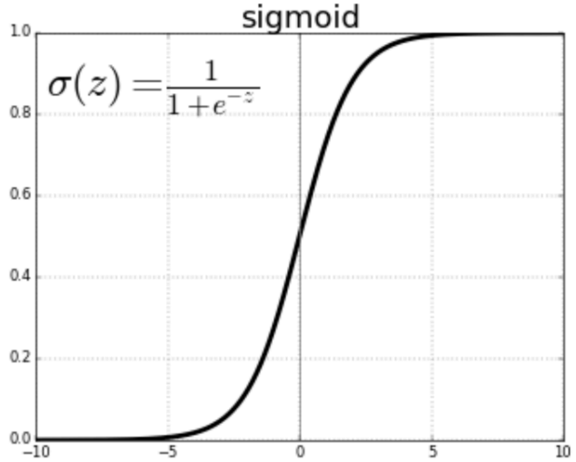
$$\mathcal{L}(\theta; Y, X) = \prod_j p_\theta(y_j|x_j)$$

  - Now we can apply this to linear classification

Logistic Regression: Conditional Distribution

- Notation: $$\sigma(z) = \frac{1}{1+\exp(-z)} = \frac{\exp(z)}{1+\exp(z)}$$

Sigmoid/ logistic



- **Conditional Distribution**:

$$P_\theta(y = 1|x) = \sigma(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$$

Logistic Regression: Loss

- Conditional MLE:

$$\log \text{ likelihood}(\theta | x^{(i)}, y^{(i)}) = \log P_\theta(y^{(i)} | x^{(i)})$$

- So:

$$\min_\theta \ell(f_\theta) = \min_\theta -\frac{1}{n} \sum_{i=1}^{n} \log P_\theta(y^{(i)} | x^{(i)})$$

Or,

$$\min_\theta -\frac{1}{n} \sum_{y^{(i)}=1} \log \sigma(\theta^T x^{(i)}) - \frac{1}{n} \sum_{y^{(i)}=0} \log(1 - \sigma(\theta^T x^{(i)}))$$

Logistic Regression: Sigmoid Properties

- **Bounded**:

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \in (0, 1)$$

- **Symmetric**:

$$1 - \sigma(z) = \frac{\exp(-z)}{1 + \exp(-z)} = \frac{1}{\exp(z) + 1} = \sigma(-z)$$

- **Gradient**:

$$\sigma'(z) = \frac{\exp(-z)}{(1 + \exp(-z))^2} = \sigma(z)(1 - (\sigma(z))$$

Logistic regression: Summary

- **Logistic regression = sigmoid conditional distribution + MLE**

- More precisely:
  - Give training data iid from some distribution D,
  - **Train**:

$$\min_{\theta} \ell(f_\theta) = \min_{\theta} -\frac{1}{n}\sum_{i=1}^{n} \log P_\theta(y^{(i)}|x^{(i)})$$

  - **Test**: output label probabilities

$$P_\theta(y=1|x) = \sigma(\theta^T x) = \frac{1}{1+\exp(-\theta^T x)}$$

Logistic Regression: Comparisons

- Recall the first attempt:

$$\ell(f_\theta) = \frac{1}{m}\sum_{i=1}^{m} 1\{\text{step}(f_\theta(x^{(i)}) \neq y^{(i)})$$

Difficult to optimize!!

Logistic Regression: Comparisons

- What if we run least squares linear regression?

$$\ell(f_\theta) = \frac{1}{n}\sum_{j=1}^{n}(f_\theta(x^{(j)}) - y^{(j)})^2$$

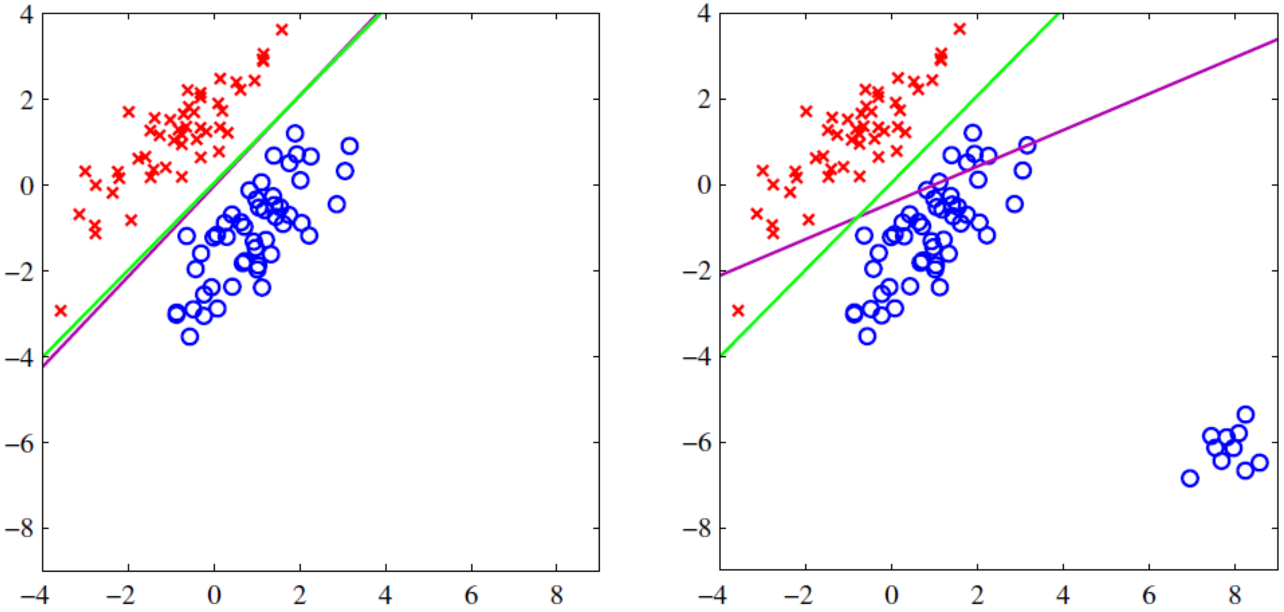


Figure 4.4 The left plot shows data from two classes, denoted by red crosses and blue circles, together with the decision boundary found by least squares (magenta curve) and also by the logistic regression model (green curve), which is discussed later in Section 4.3.2. The right-hand plot shows the corresponding results obtained when extra data points are added at the bottom left of the diagram, showing that least squares is highly sensitive to outliers, unlike logistic regression.

Figure: *Pattern Recognition and Machine Learning*, Bishop

Break & Quiz

Q3-1: Select the correct option.

A. *For logistic regression, sometimes gradient descent will converge to a local minimum (and fail to find the global minimum).*

B. *The cost function for logistic regression trained with 1 or more examples is always greater than or equal to zero.*

$$\min_{\theta} \ell(f_\theta) = \min_{\theta} -\frac{1}{n}\sum_{i=1}^{n} \log P_\theta(y^{(i)}|x^{(i)})$$

1. Both statements are true.
2. Both statements are false.
3. Statement A is true, Statement B is false.
4. Statement B is true, Statement A is false.

Q3-1: Select the correct option.

A. *For logistic regression, sometimes gradient descent will converge to a local minimum (and fail to find the global minimum).*

B. *The cost function for logistic regression trained with 1 or more examples is always greater than or equal to zero.*

1. Both statements are true.
2. Both statements are false.
3. Statement A is true, Statement B is false.
4. Statement B is true, Statement A is false.

The cost function for logistic regression is convex, so gradient descent will always converge to the global minimum.

The cost for any example is always >= 0 since it is the negative log of a quantity less than one. The cost function is a summation over the cost for each sample, so the cost function itself must be greater than or equal to zero.

Outline

- **Linear Regression**
  - Gradient-descent based solutions
- **Logistic Regression**
  - Maximum likelihood estimation, setup, comparisons
- **Logistic Regression: Multiclass**
  - Extending to multiclass, softmax, cross-entropy
- **Gradient Descent & SGD**
  - Convergence proof for GD, introduction to SGD

Logistic Regression: Beyond Binary

- We started with this conditional distribution:

$$P_\theta(y = 1|x) = \sigma(\theta^T x) = \frac{1}{1 + \exp(-\theta^T x)}$$

- Now let us try to extend it.
  - Can no longer just use one $\theta^T x$
  - But we can try multiple…

Logistic Regression: Beyond Binary

- Let's set, for y in 1,2,…,k

$$P_\theta(y = i|x) = \frac{\exp((\theta^i)^T x)}{\sum_{j=1}^k \exp((\theta^j)^T x)}$$

- Note: we have several weight vectors now (1 per class).
- To train, same as before (just more weight vectors).

$$\min_\theta -\frac{1}{n}\sum_{i=1}^n \log P_\theta(y^{(i)}|x^{(i)})$$

Cross-Entropy Loss

- Let us define $q^{(i)}$ as the one-hot vector for the ith datapoint.
- Next, let's let $p^{(i)} = P_\theta(y|x^{(i)})$ be our prediction

Note: only 1 term non-zero.

- Our loss terms can be written

$$-\log p(y^{(i)}|x^{(i)}) = -\sum_{j=1}^{k} \underbrace{q_j^{(i)} \log p(y=j|x^{(i)})}_{\text{Looks like the entropy, but ...}}$$

- This is the "cross-entropy" $H(q^{(i)}, p^{(i)})$

Cross-Entropy Loss

- This is the "cross-entropy"

$$H(q^{(i)}, p^{(i)}) = \mathbb{E}_{q^{(i)}}[\log p^{(i)}]$$

- What are we doing when we minimize the cross-entropy?
- Recall KL divergence,

$$D(q^{(i)}||p^{(i)}) = \underbrace{\mathbb{E}_{q^{(i)}}[\log p^{(i)}]}_{\text{Cross-entropy}} - \underbrace{\mathbb{E}_{q^{(i)}}[\log q^{(i)}]}_{\text{Entropy } H(q^{(i)}) \text{ (fixed)}}$$

- Matching distributions!

Softmax

- We wrote

$$P_\theta(y = i|x) = \frac{\exp((\theta^i)^T x)}{\sum_{j=1}^k \exp((\theta^j)^T x)}$$

- This operation is called softmax.
  - Converts a vector into a probability vector (note normalization).
  - If one component in the vector **a** is **dominant**, softmax(**a**) is close to one-hot vector

Quiz (do at home)

Q: Calculate the softmax of (1, 2, 3, 4, 5).

1. (0.067, 0.133, 0.2, 0.267, 0.333)
2. (0, 0.145, 0.229, 0.290, 0.336)
3. (0.012, 0.032, 0.086, 0.234, 0.636)
4. (0.636, 0.234, 0.086, 0.032, 0.012)

Q: Calculate the softmax of (1, 2, 3, 4, 5).

1. (0.067, 0.133, 0.2, 0.267, 0.333)
2. (0, 0.145, 0.229, 0.290, 0.336)
3. (0.012, 0.032, 0.086, 0.234, 0.636)
4. (0.636, 0.234, 0.086, 0.032, 0.012)

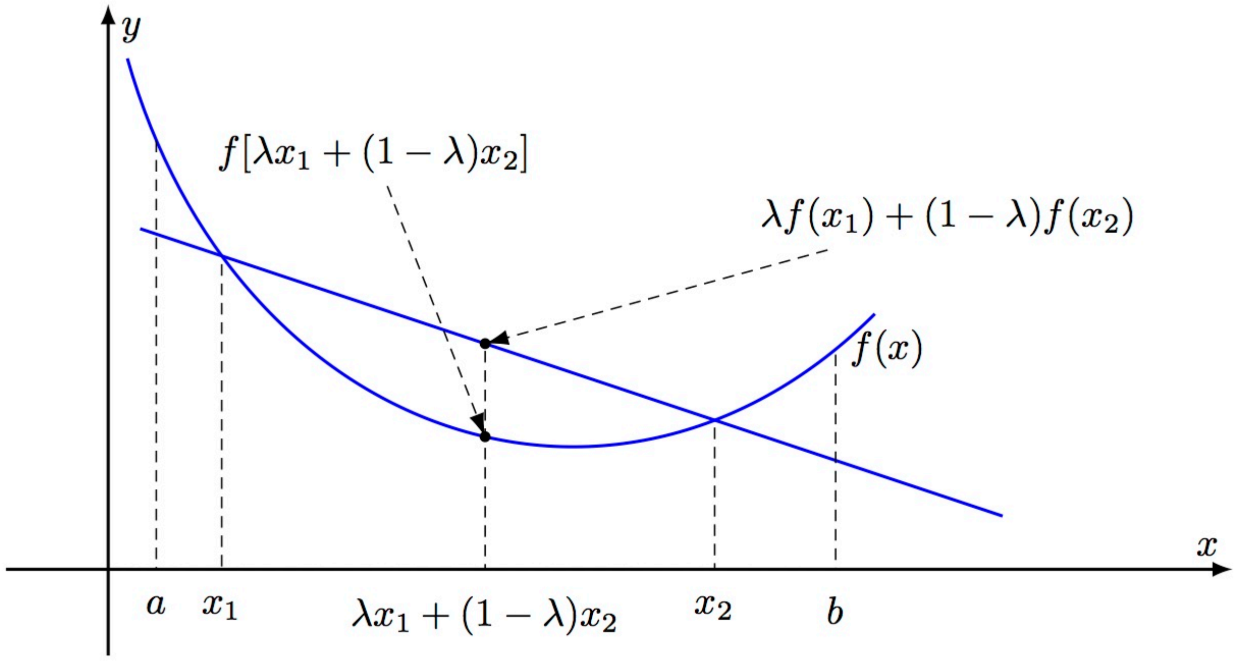
Outline

- **Linear Regression**
  - Gradient-descent based solutions
- **Logistic Regression**
  - Maximum likelihood estimation, setup, comparisons
- **Logistic Regression: Multiclass**
  - Extending to multiclass, softmax, cross-entropy
- **Gradient Descent & SGD**
  - Convergence proof for GD, introduction to SGD

Gradient Descent Analysis : Convexity

- Recall the definition of a convex function. For f, with convex domain, for all x_1, x_2 in this domain and all $\lambda \in [0, 1]$

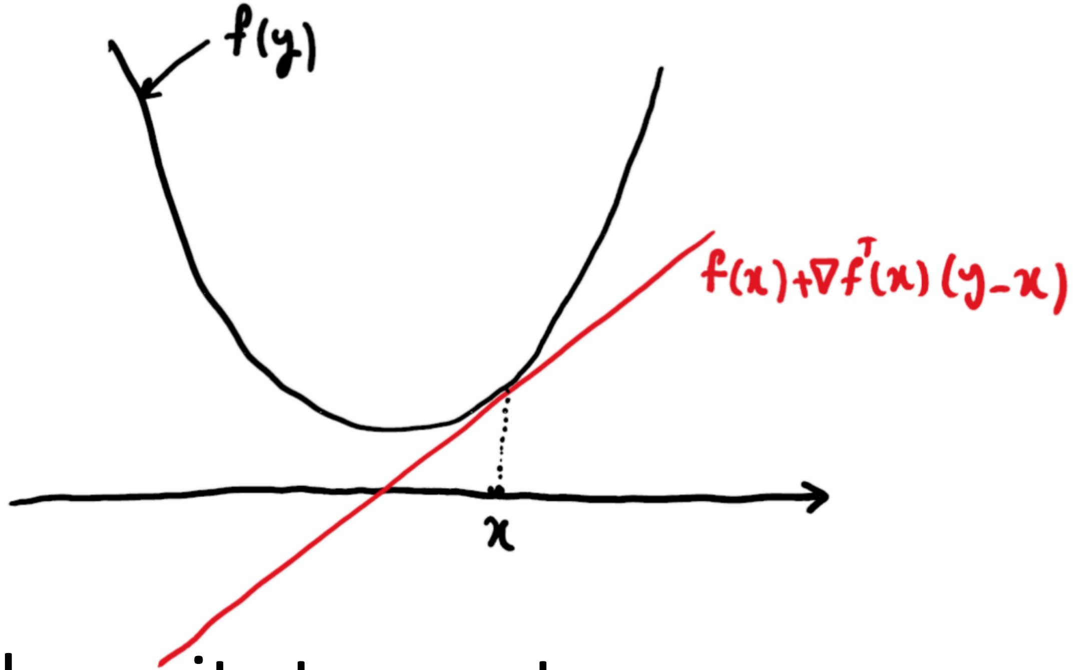
$$\underbrace{f(\lambda x_1 + (1-\lambda)x_2)}_{\text{Convex combination}} \leq \underbrace{\lambda f(x_1) + (1-\lambda) f(x_2)}_{\text{Line segment joining } f(x_1) \text{ and } f(x_2)}$$



Gradient Descent Analysis : Convexity

- An equivalent definition if f is differentiable:

$$f(y) \geq f(x) + \nabla f(x)^T (y - x)$$



- Function sits above its tangents

Gradient Descent Analysis : Lipschitzness

- Assume $\|\nabla f(x_1) - \nabla f(x_2)\|_2 \leq L\|x_1 - x_2\|_2$
  This is equivalent to

$$\nabla^2 f(x) \preceq LI$$

- Recall: $A \preceq B$ means that $B - A$ is positive semidefinite

- Recall some more: C is positive semidefinite if for all x,

$$x^T C x \geq 0$$

Gradient Descent: Convergence Proof p. 1

- Let us start with a Taylor expansion:

$$f(y) = f(x) + \nabla f(x)^T (y - x) + 1/2 (y - x)^T \nabla^2 f(z)(y - x)$$

here z is a point on the line segment between x and y.

- Next, our gradient Lipschitz condition means $\nabla^2 f(x) \preceq LI$

$$\boxed{\implies f(y) \leq f(x) + \underbrace{\nabla f(x)^T (y - x)}_{\text{Linear Approximation}} + \underbrace{1/2 L \|y - x\|^2}_{\text{Remainder: at most a quadratic}}}$$

Gradient Descent: Convergence Proof p. 2

- Let's plug in our GD relationship $y \leftarrow x_{t+1} = x_t - \alpha \nabla f(x_t)$

$$\boxed{\implies f(y) \leq f(x) + \nabla f(x)^T (y - x) + 1/2L\|y - x\|^2}$$

- Start with some algebra

$$f(x_{t+1}) \leq f(x_t) + \nabla f(x_t)^T (x_{t+1} - x_t) + 1/2L\|x_{t+1} - x_t\|_2^2$$

$$= f(x_t) - \nabla f(x_t)^T \alpha \nabla f(x_t) + 1/2L\|\alpha \nabla f(x_t)\|_2^2$$

$$= f(x_t) - \alpha\|\nabla f(x_t)\|_2^2 + 1/2L\alpha^2\|\nabla f(x_t)\|_2^2$$

$$\boxed{= f(x_t) - \alpha(1 - 1/2L\alpha)\|\nabla f(x_t)\|_2^2}$$

Gradient Descent: Convergence Proof p. 3

- Taking $\alpha = \frac{1}{L} \implies \alpha(1 - 1/2L\alpha) = \alpha/2$

- So we now have

$$\boxed{f(x_{t+1}) \leq f(x_t) - 1/2\alpha \underbrace{\|\nabla f(x_t)\|_2^2}_{\text{Positive except at minimum (where it's 0)}}}$$

We have shown that with an appropriate step size, the objective will always decrease

Gradient Descent: Convergence Proof p. 4

- Have not used convexity yet:

$$\boxed{f(x_t) \leq f(x^*) + \nabla f(x)^T (x_t - x^*)}$$

- Combine with $\boxed{f(x_{t+1}) \leq f(x_t) - 1/2\alpha \|\nabla f(x_t)\|_2^2}$

$$f(x_{t+1}) \leq f(x^*) + \nabla f(x_t)^T (x_t - x^*) - \alpha/2 \|\nabla f(x_t)\|_2^2$$

$$f(x_{t+1}) - f(x^*) \leq \frac{1}{2\alpha}(2\alpha \nabla f(x_t)^T (x_t - x^*) - \alpha^2 \|\nabla f(x_t)\|_2^2)$$

$$\boxed{f(x_{t+1}) - f(x^*) \leq \frac{1}{2\alpha}(\|x_t - x^*\|_2^2 - \|x_t - \alpha \nabla f(x_t) - x^*\|_2^2)}$$

Gradient Descent: Convergence Proof p. 5

- Now, simplify

$$\boxed{f(x_{t+1}) - f(x^*) \leq \frac{1}{2\alpha}(\|x_t - x^*\|_2^2 - \|\underbrace{x_t - \alpha \nabla f(x_t)}_{\text{This part is just } x_{t+1}} - x^*\|_2^2)}$$

$$\boxed{f(x_{t+1}) - f(x^*) \leq \frac{1}{2\alpha}(\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2)}$$

Gradient Descent: Convergence Proof p. 6

- With the following bound,

$$\boxed{f(x_{t+1}) - f(x^*) \leq \frac{1}{2\alpha}\Big(\underbrace{\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2}_{\text{Can telescope if we sum over t!}}\Big)}$$

$$\sum_{t=0}^{T-1} f(x_{t+1}) - f(x^*) \leq \sum_{t=0}^{T-1} \frac{1}{2\alpha}(\|x_t - x^*\|_2^2 - \|x_{t+1} - x^*\|_2^2)$$

$$\boxed{\sum_{t=0}^{T-1} f(x_{t+1}) - f(x^*) \leq \frac{1}{2\alpha}(\|x_0 - x^*\|_2^2 - \|x_T - x^*\|_2^2)}$$

Gradient Descent: Convergence Proof p. 7

- Now we have

$$\sum_{t=0}^{T-1} f(x_{t+1}) - f(x^*) \leq \frac{1}{2\alpha}(\|x_0 - x^*\|_2^2 - \|x_T - x^*\|_2^2)$$

- Can ignore the rightmost term (we're just making the RHS same or bigger)

$$\sum_{t=0}^{T-1} \underbrace{f(x_{t+1}) - f(x^*)}_{\text{Value gap for all steps}} \leq \underbrace{\frac{1}{2\alpha}(\|x_0 - x^*\|_2^2)}_{\text{Initial guess gap to minimizer}}$$

Gradient Descent: Convergence Proof p. 7

- Continue,

$$\sum_{t=0}^{T-1} f(x_{t+1}) - f(x^*) \leq \frac{1}{2\alpha}(\|x_0 - x^*\|_2^2)$$

- But, recall that each iterate has a smaller value, ie,

$$f(x_{t+1}) \leq f(x_t) - 1/2\alpha\|\nabla f(x_t)\|_2^2$$

- So,

$$\sum_{t=0}^{T-1} f(x_T) \leq \sum_{t=0}^{T-1} f(x_{t+1})$$

Gradient Descent: Convergence Proof p. 8

- We have

$$\sum_{t=0}^{T-1} f(x_T) \leq \sum_{t=0}^{T-1} f(x_{t+1})$$

- Divide by T,

$$f(x_T) - f(x^*) \leq \frac{1}{T} \sum_{i=0}^{T-1} f(x_t) - f(x^*)$$

- Combine with

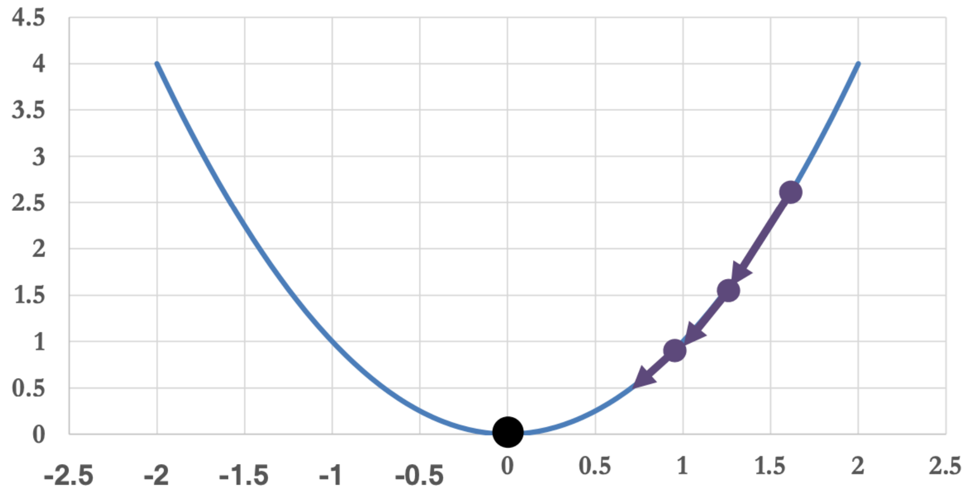
$$\sum_{t=0}^{T-1} f(x_{t+1}) - f(x^*) \leq \frac{1}{2\alpha}(\|x_0 - x^*\|_2^2)$$

$$\boxed{\implies f(x_T) - f(x^*) \leq \frac{\|x_0 - x^*\|_2^2}{2T\alpha}}$$

Done!

Gradient Descent: Some notes on the proof

- Proof credit: Ryan Tibshirani (CMU).
- Other assumptions that lead to varying proofs/rates:
  - **Strong convexity**
  - **Non-convexity**
  - **Non-differentiability**



Gradient descent: Downside

- Why would we not use GD?

- Let's go back to ERM. $$\arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \ell(h(x^{(i)}, y^{(i)})$$

- For GD, need to compute $\nabla \ell(h(x^{(i)}, y^{(i)})$

  - Each step: n gradient computations
  - ImageNet: 10^6 samples… so for 100 iterations, **10^8 gradients**

Solution: **Stochastic** Gradient Descent

- Simple modification to GD.
- Let's use some notation: ERM:

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\theta; x^{(i)}), y^{(i)})$$

Note: this is what we're optimizing over! x's are fixed samples.

- GD: $$\theta_{t+1} = \theta_t - \frac{\alpha}{n} \sum_{i=1}^{n} \nabla \ell(f(\theta_t; x^{(i)}), y^{(i)})$$

Solution: **Stochastic** Gradient Descent

- Simple modification to GD:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{n} \sum_{i=1}^{n} \nabla \ell(f(\theta_t; x^{(i)}), y^{(i)})$$

- SGD:

$$\theta_{t+1} = \theta_t - \alpha \nabla \ell(f(\theta_t; x^{(a)}), y^{(a)})$$

  - Here, a is selected uniformly from 1,…,n ("**stochastic**" bit)
  - Note: **no sum**!
  - In expectation, same as GD.

Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov, and Fred Sala